Visualization of an Atmospheric Dataset

Adria Liszka
12 August 1999
Brookhaven National Laboratory

United States Department of Energy
Office of Biological and Environmental Research
Global Change Education Program
Summer Undergraduate Research Experience

ABSTRACT

Due to an almost unmanageable size, atmospheric datasets necessitate the use a clear and logical analysis tool: visualization. With this tool, scientists can study complex data in the form of images, and develop hypotheses based on the patterns of data. This project concentrates on taking the first step towards a comprehensive and automated atmospheric visualization program. The data parameters within the sets, such as those from the European Center for Medium-Range Weather Forecasts (ECMWF), fall into one of four categories: two-dimensional scalars, three-dimensional scalars, two-dimensional vectors, and three-dimensional vectors. Since the two-dimensional scalar visualization task has already been tackled, the project focuses on adding data representation capability in the other categories. This goal is accomplished through the programs `tempanim.net` (three-dimensional scalar data temperature), `surfacewindanim.net` (two-dimensional vector data surface wind) and `windanim.net` (three-dimensional vector data wind) in IBM's Data Explorer. These simplified programs render only one type of data because the overall goal of complete generalization is too complicated to achieve immediately. This paper focuses on the accomplishments of the three programs and discusses the next steps towards the ideal, complete data visualization program of the future.

INTRODUCTION

A meteorological dataset from the European Center for Medium Range Weather Forecasting has 76 parameters measured at fifteen different times on a mesh of 360 degrees longitude, 162 degrees latitude and 27 meters in height. The dataset holds almost two billion points. After dividing the data into two hemispheres, each parameter has approximately 12 million points, and each parameter's time slice about three-quarters of a million points. Almost every way of approaching this complex dataset, then, is a daunting task.

Data visualization is one of the simple, inherent tools one can use to interpret such large, complex, heterogeneous datasets. The human perceptual system has a transfer rate of about two gigabytes per second, and can determine position, color, depth, texture, shape, movement, and pattern quite well.[1] This characteristic makes visualization a leading data analysis technique: the user can see the data and its changes, and explore the relationships between these two states.

This project combines the data of `ecmwfup95110106-95110418.dat` with the realization resources available through IBM's Data Explorer. The three programs discussed, three-dimensional scalar, two-dimensional vector, and three-dimensional vector, each visualize example parameters. This paper includes several aspects of each program: a general overview, a description of the program's pages, an outline of the data flow through the program, a summary of the options available to the user, a listing of the files needed to run each program, a selection of pre-rendered examples for the user to view, and a discussion of future directions for the program. Other than individual program information, the paper focuses on providing the user with a background to the data and

[1] Gahegan, 1999.

the programming environment, suggesting some future directions for the project in general, and drawing conclusions on the visualization work accomplished so far.

BACKGROUND

In order to construct a functional visualization, one must recognize that there are two components of the problem. There is the data, the science, and the realization, the programming environment. Most importantly, however, an effective visualization must marry these two realms. This section briefly discusses the two components that go into the visualization: the science and the programming environment.

The science of this project is captured in meteorological dataset from the European Center for Medium-Range Weather Forecasts (ECMWF). All values were taken from the initialized analysis surface and model level fields except for the precipitation, heat fluxes, and thermal radiation values which were taken from the forecast surface fields at steps 12, 18, 24, 30, and 36 hours to minimize the effects of the ECMWF model spinup.[2] With a resolution of 1^{o}, the data ranges from 0^{o} to 360^{o} longitude and from the equator to 81^{o}S (ACE-1) or to 81^{o}N (ACE-2). The data also extends from the surface to 100 hPa in 27 discrete levels.

The programming environment is an essential aspect of the visualization, since it is through this program that the data is finally visualized. A desirable environment is not only complete and flexible, in order to fulfill the demands of the programmer, but also simple and logical, in order to be accessible to the non-programmers that will use it. This project used IBM's Data Explorer. Data Explorer imports, alters, and renders the data through the use of its modules, which are linked in a visually logical way. To become acquainted with the program, watch the tutorial tape then browse through the sample programs in `/usr/lpp/dx/samples/programs` and `/usr/lpp/dx/samples/tutorial`. These resources explain nearly every module or procedure. For guidance while programming, refer to Data Explorer’s User’s Guide or the User's Reference.

MODELING SCALAR DATA

General Program Description

This program imports a scalar data set (parameter 130) then renders this data as a volume or an isosurface. The user can scale the data along the z-axis to spread out the levels. In order to orient the user, there is a base map and a set of axes which accompanies each image. In addition, the image or a sequence of images can be written to a file.

Page Description

There are seven pages to the program: **ReadMe**, **Start**, **Finish**, **Volume**, **Isosurface**, **Little_Proc**, and **Axes**. The first page, **ReadMe**, describes all of the components of the

[2] Fielding, 1998.

program. **Start** imports the data, selects the rendering sequence (single frame or animation), and flips the y- and z-axes. **Finish** writes the data, the caption, the color bar legend and the map to the image, and writes this image to a file if the user wishes. It also selects the type of rendering (volume or isosurface). **Volume** and **Isosurface** renders the data in their respective formats with a scale along the z-axis. **Isosurface** also includes an *Integer* option that selects the number of isosurfaces to be rendered. **Little_Proc** gives the color scheme for the renderings. **Axes** adds an orientation for the user in the form of a three-dimensional axis (longitude, latitude, and height).

Data Flow

The program begins at **Start**, where *Selector* points to a single frame rendering or animation. The data is then imported by frame with the frame number, an essential part of the image's filename. Because the meteorology data is written 0-360, 81-0, 27-0 on the x, y, z axes respectively, the y and z positions flipped. The resulting data goes to *Import_data*.

There are two *Receivers* for *Import_data*, one on **Volume** and one on **Isosurface**. Both pages receive the data, scale it, color it, and send it to a *Switch* on **Finish**. **Isosurface** has an additional component, an *Integer*, which allows the user to select the number of isosurfaces he or she wants to see.

On these pages, the data is colored by a *Receiver* that originates in **Little_Proc**. The aim of the procedure is to color the data set so that it is intrinsic to the user. The control points in hue are (320, 0), (300, 0), (285, 0.05) and (180, 0.8).[3] The data between 300 K and 320 K (27-47 ^{o}C or 81-117 ^{o}F) is red (hue=0) because this range is considered 'hot' for a human. Some distinction (hue=0.05) is provided at 285 K (12 ^{o}C, 54 ^{o}F), which is a 'warm' red-orange. From this point there is a continuous ramp to the final value. The final control point, at 185 K (-93 ^{o}C, -135 ^{o}F), is violet (hue=0.8), which accentuates the cold temperatures much better than a deep blue.

The *Receivers Volume* and *Isosurface* both plug into *Switch* on **Finish**, but only one can be chosen by the user to render data. *Image*, then, consists of a data rendering, a *Caption*, a *Colorbar* and a map (*ReadImage*). If the user clicks the *Toggle* for *WriteImage*, the program places the image in a file (`image.`*`frame_number`*`.tiff`) in the main directory.[4]

Options

In the control panel *Scalar 3D Control Panel*, the user has six options: Write Image, Type of Rendering, Number of Isosurfaces, Z-Scale, Frame Sequence and Frame. Write Image places the frame(s) in a file in the main directory. With Type of Rendering, a *Selector*, the user can choose between seeing a volume or isosurface image. If the user chooses isosurface rendering, he or she must also alter Number of Isosurfaces, an *Integer*. The next option, Z-Scale (*Integer*), multiplies the z position in order to spread out the data. Frame Sequence, a *Selector*, allows the user to view a single frame or animation. If

[3] The control points are given in this paper as (*data value*, *hue value*).
[4] The main directory is referred to throughout the course of this paper. It is found on **`mary`**, under `/usr2/people/aliszka`.

the user wants to see a single frame, then he or she must also alter Frame, which is an *Integer* option giving the frame to view.

Necessary Files

The following files, listed here in the main directory, are needed to run this program. Where a `net` file is listed, the associated `cfg` is required also.

- `meteo/scalar3Danim/tempanim.net`
  Name of the program.
- `headers/newtemp/timetemp.dx`
  Import for this is on **Start**. Holds labels which become dates in caption.
- `headers/newtemp/vnitemp.dx`
  Import for this is on **Start**. Holds filenames in animation sequence.
- `meteo/scalar3Ddata/geo.filter`
  Filter for this on **Start**. Creates header for data file.
- `meteo/scalar3Ddata/geo.general`
  Import for these is on **Axes**.
  Creates axes that do not shift during course of animation.
- `meteo/scalar3Ddata/temp95110106.dat`
  Called from file `headers/newtemp/vnitemp.dx`. Data file. Other data files (last 3 digits only): 112, 118, 200, 206, 212, 218, 300, 306, 312, 318, 400, 406, 412, 418.
- `maps/S360.gif`
  ReadImage for this is on **Finish**. Puts base map into scene.

Examples

In each of these files lies an example of the program. The first letter of the filename stands for the type of rendering (I=isosurface, V=volume). If it is an I, then it is followed by the number of isosurfaces. After the underscore is a string describing the view, which is diagonal, off diagonal, off bottom or off front. If there is no string given then the view is a hybrid or unknown. After the next underscore, the number after z is the scaling along the z-axis. No number means that this is unknown. Finally, the extension `tiff` is an image (found in the main directory under `examples/images/scalar3D`) and `mov` is a movie (found in the main directory under `examples/movies/scalar3D`).

```
I12_diag_z1.tiff     V_offbot_z3.tiff     I10_z3.mov
I12_offbot_z3.tiff   V_offdiag_z5.tiff    I12.mov
I4_offdiag_z5.tiff   V_offfront_z3.tiff   V_offbot.mov
V_diag_z1.tiff       I6.mov               V_SW.mov
```

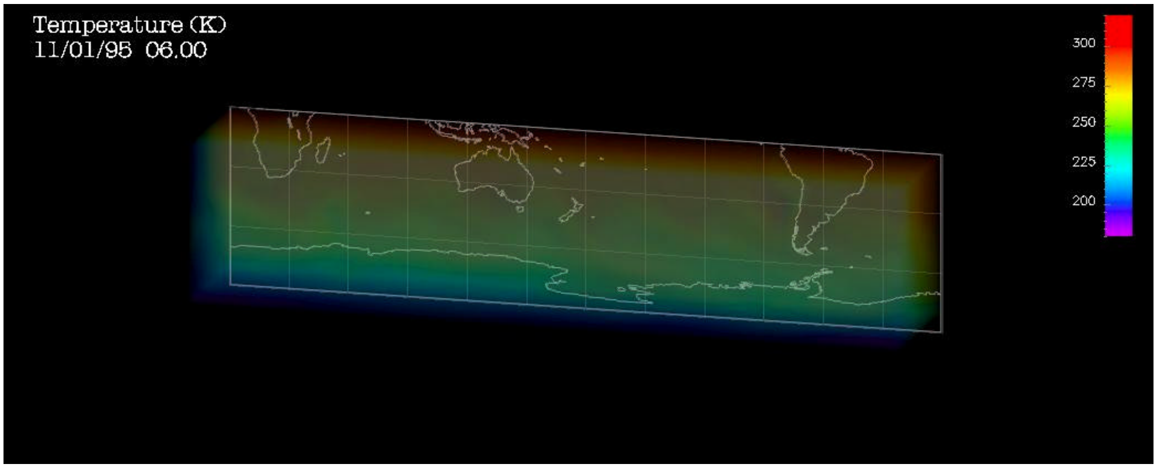


Image 1. *This image, seen in an off-front view, is a volume-rendered temperature file data with a magnification of 3 along the z-axis. It is listed above as* `V_offfront_z3.tiff`.

Future Directions

The main directions for this program take one of two forms. The first corrects current glitches in the program, while the second makes the program run more smoothly or be more user-friendly.

There are two problems with the current program. The first, which concerns the *AutoAxes*, is most significant. With the *AutoAxes* plugged into *Collect* on **Finish**, the program runs slowly if at all. Even setting the axes through VIEW CONTROL in Data Explorer does not work.

The second problem pertains to the color of the resulting volume animations. In a movie, the volume images look a bit segmented or layered, not as smooth as the images themselves or a movie viewed through Data Explorer. One solution to this problem may be to choose another movie player. The best solution, however, would be to render the images more quickly so that the movie can be played directly from Data Explorer.

MODELING VECTOR DATA IN TWO DIMENSIONS

General Program Description

This program imports the two components of surface wind (parameters 131 and 132) as one-dimensional vectors, combines them in a two-dimensional vector, then allows for the rendering of this data in several ways. This includes arrow glyphs, text glyphs, streamlines, ribbons, and a contour map of the vector magnitudes. In addition, the user can animate the images, write them to a file, and add an underlying map.

Page Description

There are eight pages to the program: **ReadMe**, **Start**, **Finish**, **Axes**, **Chooser**, **Sampling**, **Options**, and **Color**. The first page is **ReadMe**, which describes all of the components of the program. **Start** imports the data, selects the rendering sequence (single frame or animation), flips the y-axis and slices the data. **Finish** writes the data, the caption, the color bar legend and the map to the image, and writes this image to a file if the user wishes. **Axes** imports the data from the first frame and constructs a set of axes from it. **Chooser** holds the *Selector* and *Switch* that allow for several rendering options: text glyphs, arrows, streamlines, ribbons, a contour map, or no data rendering. **Sampling** computes the size of the sampled field from two input *Integers*. **Options** allows the reader to turn on (or off) the base map. **Color** gives the color scheme for the renderings.

Data Flow

The program begins at **Start**, where *Selector* points to a single frame rendering or an animation. The u and v components of wind are then imported separately along with the frame number, which is a crucial element of the image's filename. The two components (u and v) are then collected in a single vector and the data gets fed to *Stream*. The y position (latitude) is flipped because the data is originally written 81-0. One layer of the data (*Slice*) is taken so that only surface wind shows.

Stream then goes into **Color** and **Chooser**. **Color** is a simple page that colors the data and outputs to a *Colorbar,* or colored legend. Three colors span the range of data, from 0-100 m/s. The control points in hue are (100, 0.18), (60, 0.18), (60, 0.62), (40, 0.62), (40, 0) and (0, 0). The colors are blocked (there is no gradient between them) as a multitude of colors tends to overshadow the vector's direction, a most significant component. The blocks were chosen through a statistical analysis of the data, which showed the mean to be approximately 16 m/s and the standard deviation to be about 12 m/s. Most of the data, then, is concentrated at one end, so this data is red (hue=0). This color shows up well on either a white or black background. The upper level of the data, 60-100 m/s, had very few values, but nevertheless they need to be visible. For this reason, they are colored yellow (hue=0.18). Finally, the transition data, not as extreme as 60-100 m/s but significantly higher (two standard deviations) than the rest of the data, is a middle blue. Unlike other colors, this blue (hue=0.62) looks very different from the red and yellow, but does not blend in to the white map outlines or the black background.

From **Color**, the data goes into **Sampling**, where user input reduces the number of elements in the data field. On **Chooser**, the other page which accepts the *Stream* data, the *Switch* picks one of the many rendering options to send to the final page. This page also has a *Receiver* from **Sampling**, which is called *Data*, and a *Compute* that performs the final calculations for the contour map.

Data also flows through two independent pages: **Axes** and **Options**. **Axes** adds axis annotation for every image, but uses only the data set from the first day so the axes do not shift in animation. **Options** imports the base map.

Many *Receivers* plug into the *Collect* on **Finish**, then the resulting *Image* is displayed. If the user clicks on *Toggle* for *WriteImage*, the program places the image in a file (`image.`*`frame_number`*`.tiff`) in the main directory.

Options

In the control panel *Vector 2D Control Panel*, the user has seven options: Data Representation, X Scale, Y Scale, Write Image, Show Map, Frame Rendering and Frame. With Data Representation, a *Selector*, the user can choose between text glyphs, arrows, streamlines, ribbons, a contour map, or no representation. The option to show no data was added so that the user can view the base map alone. The next two options, X Scale and Y Scale, are both *Integers* that reduce the number of data points in the given directions. A '1' in both of these fields produces an image with no sampling. Write Image and Show Map are both *Toggle* options. Write Image places the frame(s) in a file, while Show Map draws a base map in the scene. Frame Rendering, a *Selector*, allows the user to view a single frame or an animation. If the user wants to see a single frame, then he or she must also alter Frame, which is an *Integer* option giving the frame to view.

Necessary Files

The following files, listed here in the main directory, are needed to run this program. Where a `net` file is listed, the associated `cfg` is required also.

· `meteo/vector2Danim/surfacewindanim.net`
  Name of the program.
· `headers/newwind/timewind.dx`
  Import for this is on **Start**. Holds labels which become dates in caption.
· `headers/newwind/vniUwind.dx`
· `headers/newwind/vniVwind.dx`
  Import for this is on **Start**. Holds filenames in animation sequence.
· `meteo/scalar3Ddata/geo.filter`
  Filter for this on **Start**. Creates header for data file.
· `meteo/vectordata/uwind0106.general`
· `meteo/vectordata/vwind0106.general`
  Import for these is on **Axes**.
  Creates axes that do not shift during course of animation.
· `meteo/vectordata/uwind95110106.dat`
· `meteo/vectordata/vwind95110106.dat`
  Called from file `headers/newwind/vniUwind.dx`, `headers/newwind/vniVwind.dx`.
  Data file. Other data files (last 3 digits only): 112, 118, 200, 206, 212, 218, 300, 306, 312, 318, 400, 406, 412, 418.
· `maps/S360.gif`
  ReadImage for this is on **Options**. Puts base map into scene.

Examples

In each of these files lies an example of the program. The letter of the filename stands for the type of rendering (A=arrow, C=contour map, N=none, R=ribbon, S=streamline, T=text glyph). The first three numbers correspond to the sampling in the x (longitudinal) direction (max 360), and the next two numbers refer to the sampling in the y (latitudinal) direction (max 81). For ribbons, the final number is the thickness of the ribbon. If there are no numbers then sampling does not play a part in rendering the image. The extension `tiff` is an image (found in the main directory under `examples/images/vector2D`) and `mov` is a movie (found in the main directory under `examples/movies/vector2D`).

```
A00403.tiff     R03618_2.tiff  T12081.tiff    R03618_2.mov
A00606.tiff     S00606.tiff    A00606.mov     S01209.mov
C.tiff          S01209.tiff    A00503.mov     S03009.mov
N.tiff          S03009.tiff    C.mov          T12081.mov
```

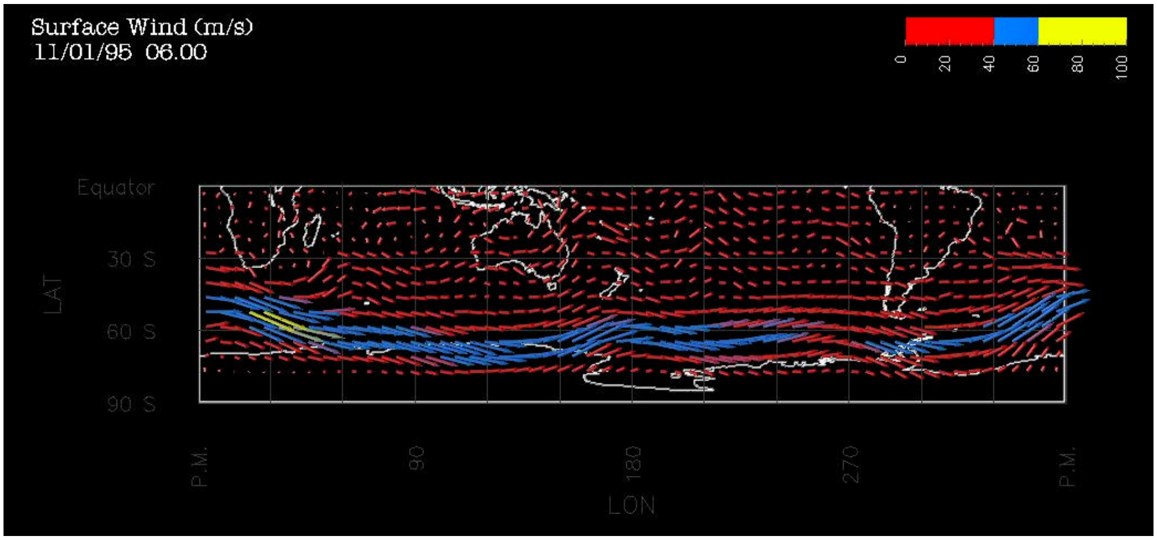


Image 2. *This surface wind image, seen in the front view, is rendered with arrow glyphs at a sampling rate of 6 in both the x and y directions. It is listed above as* `A00606.tiff`.

Future Directions

This program is limited in that is only visualizes two-dimensional vectors, so the main improvements should focus on five difficulties currently in the program. The first of these is most significant, as it appears in both the two-dimensional vector and three-dimensional vector programs. It concerns sampling, which is an option the user has in the control panel. As it is now, the sampling occurs *after* the u and v components are placed into the two-dimensional vector, so the entire array is sampled, not just the individual components. The sampling field that is generated, then, might not be the same one the user specifies. For example, a sampling request of '9,3' in the x and y directions respectively would yield the same as a request for '3,9'.[5] I attempted to sample before the u and v components were put together, but this only caused more problems.

The contour map rendering plays a role in both the second and third problems. The first of these concerns the base map. The background of the base map is black, so the contour map is hidden behind the map when the two are supposed to show together. The other defect depends upon *AutoAxes*. In order to draw the axes, the input must have a color to it. I chose black, because this color does not interfere with the other images. However, the contour map shows the position of the data as a neat array of black dots. A continuous contour map would be preferable.

The next issue involves the *Streamline* renderings (*Streamline* and *Ribbon*). Both renderings produce yellow lines. I tried to rectify this by coloring the data input to this

[5] A sampling request of '9,3' would give 360/9 * 81/3 points, or 1080 points. Due to the associative nature of multiplication, however, a request of '3,9' would give 360/3 * 81/9 points, or 1080 points as well.

module (using *From_color* instead of *Stream*), but the lines stayed yellow. It may be misleading that they are the color associated with the 60-100 m/s range. However, if using this module transforms the data in some way to place it in this range, then some explanation should be added for the user's benefit.

The final flaw concerns *Ribbon*. This rendering does not flatter the two-dimensional vector data set. Perhaps *Ribbon* should be reserved for the three-dimensional data set, and only *Streamline* should be used here.

MODELING VECTOR DATA IN THREE DIMENSIONS

General Program Description

This program imports the three components of surface wind (parameters 131, 132, 500) as one-dimensional vectors, combines them in a three-dimensional vector, then allows for the rendering of this data in several ways. This includes arrow glyphs, streamlines, ribbons, and a contour map of the vector magnitudes. In addition, the user can animate the images, write them to a file, and add an underlying map, among other options.

Page Description

There are eight pages to the program: **ReadMe**, **Start**, **Finish**, **Axes**, **Chooser**, **Sampling**, **Options**, and **Color**. The first page is **ReadMe**, which describes all of the components of the program. **Start** imports the data and selects the rendering sequence (single frame or animation). **Finish** writes the data, the caption, the color bar legend, the axes and the map to the image, and writes this image to a file if the user wishes. **Axes** imports the data from the first frame and constructs a set of axes from it. **Chooser** holds the *Selector* and *Switch* that are responsible for several rendering options: arrows, streamlines, ribbons, a contour map, or no data rendering. **Sampling** computes the size of the sampled field from two input *Integers*. **Options** allows the reader to turn on (or off) the base map, *Switch* between Northern and Southern Hemisphere data, and *Toggle* between meteorology data (from ECMWF) or no meteorology data (model data, from GChM-O). **Color** gives the color scheme, which is the same for all renderings except the contour map.

Data Flow

The program begins at **Start**, where *Selector* points to a single frame rendering or an animation. The u, v and w components of wind are then imported separately along with the frame number, which is a crucial element of the image's filename. The three components (u, v, w) are then collected in a single vector and the data gets fed to *Start_data*. To ensure every layer of the data is taken, a *Slab* is used in the z-direction before the data moves on from this page. This is where the sampling occurs in the z-direction.

Start_data then goes into **Options**. If the *Toggle* Meteorology Data (ECMWF) is chosen, the data is flipped along the y- and z-axes. This data is in the form 0^o to 360^o longitude, 81^o to 0^o degrees latitude, and level 27 to 0 in height: y and z are flipped and justified across their respective axes so that north and off the ground are both 'up.' If the meteorology data is not chosen, the user is importing the model (GChM-O) data, which does not need to be flipped. The data exiting this sequence is *flip_data*.

Flip_data is imported to two pages. On **Chooser**, it is used as an input for every rendering option except *AutoGlyph*, the arrows. *AutoGlyph* also receives the flipped data, but only after it has been processed through a coloring sequence on **Color** and a sampling sequence on **Sampling**. **Color** is a simple page that colors the data prior to rendering. There are two types of coloring schemes used here. The first scheme is used on every type of rendering except the contour map. This *Colormap* takes the sized data according to the u and v components, then colors each vector red (hue=0) if the w component is greater than zero (if the vector points 'up') and blue (hue=0.667) if the w component is less than zero (if the vector points 'down'). The second coloring scheme, for the contour map only, is even simpler. It colors the vector magnitudes on a continuous gradient from violet (hue=0.8) for the lowest value to red (hue=0) for the highest value. An extra control point (95, 0) is inserted to extend the red slightly and make the color bar more balanced.

Sampling receives the data from **Color** and also processes the sampling in the z-direction through the *Selector* and *Slab* options. In the x- and y-directions, the entire field is sampled, so the problem of association noted in the previous section still exists. where user input reduces the number of elements in the data field. On **Chooser**, the other page which accepts the *Stream* data, the *Switch* picks one of the many rendering options to send to the final page. This page also has a *Receiver* from **Sampling**, which is called *Data*, and a *Compute* that performs the final calculations for the contour map.

Data also flows through two independent procedures: **Axes** and the *ReadImage* procedure on **Options**. **Axes** adds axis annotation for every image, but uses only the data set from the first time so the axes do not shift in animation. The *ReadImage* procedure on **Options** has the *Toggle* responsible for showing the base map.

Many *Receivers* plug into the *Collect* on **Finish**, then the resulting *Image* is displayed. If the user clicks on *Toggle* for *WriteImage*, the program places the image in a file (`image.`*`frame_number`*`.tiff`) in the main directory.

Options

In the control panel *Vector 3D Control Panel*, the user has ten options: Data Representation, X Scale, Y Scale, Z Sampling, Write Image, Show Map, Frame Rendering, Frame, Hemisphere, and Meteorology Data (ECMWF). With Data Representation, a *Selector*, the user can choose between arrows, streamlines, ribbons, a volume contour map, or no representation. The option to show no data was added so that the user can view the base map alone. The next two options, X Scale and Y Scale, are both *Integers* that reduce the number of data points in the given directions. A '1' in both of these fields produces an image with no sampling. Z Sampling also reduces the data in the field, but as a *Selector* that feeds into a *Slab* module. The choices for sampling in the z-direction are 'every point,' 'every 3rd point,' and 'every 9th point.' Write Image and

Show Map are both *Toggle* options. Write Image places the frame(s) in a file, while Show Map draws a base map in the scene. Frame Rendering, a *Selector*, allows the user to view a single frame or an animation. If the user wants to see a single frame, then he or she must also alter Frame, which is an *Integer* option giving the frame to view. Hemisphere, a *Selector*, allows the user to view either Southern (ACE-1) or Northern (ACE-2) Hemisphere data. The Toggle Meteorology Data (ECMWF) allows the user to view either meteorology or model data.

Necessary Files

The following files, listed here in the main directory, are needed to run this program. Where a `net` file is listed, the associated `cfg` is required also.

- `meteo/vector3Danim/volumewindanim.net`
  Name of the program.
- `headers/newwind/timewind.dx`
  Import for this is on **Start**. Holds labels which become dates in caption.
- `headers/newwind/vniUwind.dx`
- `headers/newwind/vniVwind.dx`
- `headers/newwind/vniWwind.dx`
  Import for this is on **Start**. Holds filenames in animation sequence.
- `meteo/scalar3Ddata/geo.filter`
  Filter for this on **Start**. Creates header for data file.
- `meteo/vectordata/uwind0106.general`
- `meteo/vectordata/vwind0106.general`
- `meteo/vectordata/wwind0106.general`
  Import for these is on **Axes**.
  Creates axes that do not shift during course of animation.
- `meteo/vectordata/uwind95110106.dat`
- `meteo/vectordata/vwind95110106.dat`
- `meteo/vectordata/wwind95110106.dat`
  Called from file `headers/newwind/vniUwind.dx`, `headers/newwind/vniVwind.dx` and `headers/newwind/vniWwind.dx`. Data file. Other data files (last 3 digits only): 112, 118, 200, 206, 212, 218, 300, 306, 312, 318, 400, 406, 412, 418.
- `maps/S360.gif`
  ReadImage for this is on **Options**. Puts base map into scene.

Examples

In each of these files lies an example of the program. The letter of the filename stands for the type of rendering (A=arrow, N=none, R=ribbon, S=streamline, V=Volume contour map). After this letter is a string describing the view, which is diagonal, off diagonal, or off front. If there is no string given then the view is a hybrid or unknown. The first three numbers correspond to the sampling in the x (longitudinal) direction (max 360), and the next two numbers refer to the sampling in the y (latitudinal) direction (max 81). The number after the dash is the sampling in the z-direction. The extension `tiff` is an image (found in the main directory under `examples/images/vector2D`) and `mov` is a movie (found in the main directory under `examples/movies/vector2D`).

```
Adiag03618-9.tiff           A00909-3.mov
Aoffdiag00909-3.tiff        Aofffront00606-9.mov
```

```
Aofffront00606-9.tiff        Vdiag-9.mov
Vdiag-9.tiff
```

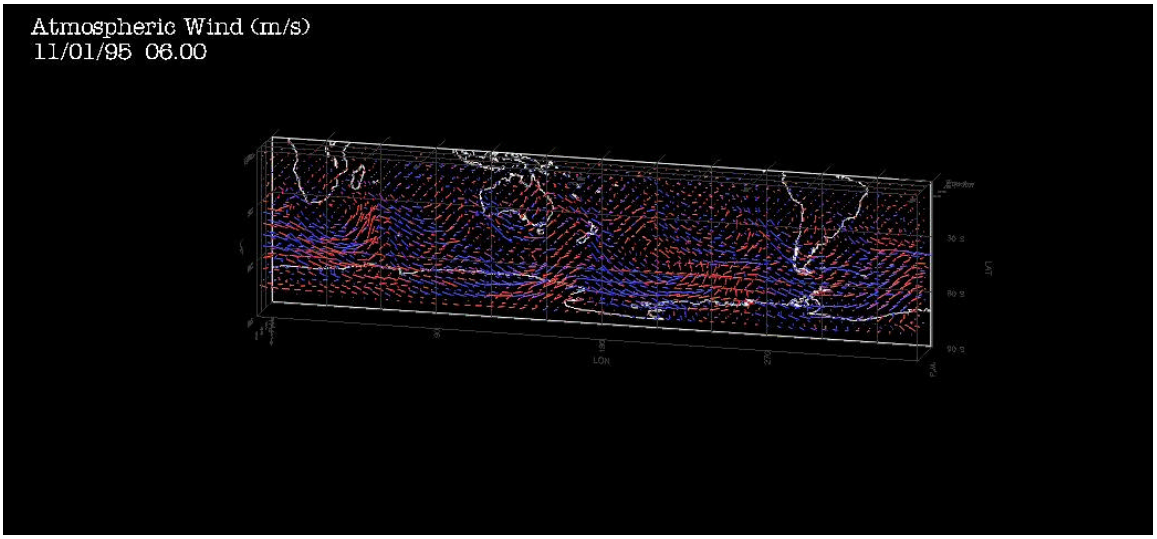


Image 3. *This wind image, seen in the off-front view, is rendered with arrow glyphs at a sampling rate of 6 in both the x and y directions and 9 in the z-direction. It is listed above as* `Aofffront00606-9.tiff`.

Future Directions

Of the three programs, this one is the most advanced, but nevertheless is has some basic problems. Correcting these problems and adding some options could greatly improve this program.

The most obvious problem is the program's inability to utilize the Streamline module, which mean that neither Streamlines nor Ribbons can be rendered. The program terminates abruptly when these options are tried, and it gives an error message. Other problems include the associative nature of the sampling procedure in the x and y directions, which was mentioned previously. This program might offer a solution, however, as it samples in the z-direction with a *Slab* and a *Selector*. The related program `/meteo/vector3Danim/windanim2.net` attempts to sample in all directions like this, but does not work. The final problem concerns the view. A new view should be assigned that allows the user to easily see all pertinent data and axes.

Most of the additions that could be added are evident. Firstly, *Scales*, similar to the Z-Scale found in the three-dimensional scalar program, should be added for all three dimensions. This would spread the data in any or all directions and make it more readable. Secondly, a control panel of *FileSelectors* would make the name of the data file more accessible to the user. When the program is more generalized, this will be a necessity. Finally, the *Selector* controlling the hemisphere should be linked into the program. A generalized version of this program would allow the user to *Switch* between the ACE-1 and ACE-2 data. The data, the map, the axes and the labels would all be affected.

FUTURE PROJECT DIRECTIONS

Generalize the programs.

This project will eventually yield an essential tool for exploration and analysis in atmospheric science. On the way to being part of this ambitious task, the three programs introduced here must be generalized. As it is now, each program renders one parameter set from the dataset `ecmwfup95110106-95110418.dat`. The list below suggests capabilities that the program of the future should possess and some ideas on fulfilling these requests.

- *The program should be able to import the entire database from ECMWF*. In order to do this, it must download files such as `ecmwfup95110106-95110418.dat` and extract the data components in a manageable way.
- *The user should be able to select a dataset, either ACE-1 or ACE-2*. The program will not only have to point to the correct subdirectory of data, but take into account the changes in the map, captions, and axes resulting from the change of hemisphere.
- *The user should be able to select the dataset type, either meteorological (ECMWF) or model (GChM-O) data*. This choice could even be encoded into the dataset name (i.e. ACE-1-ECMWF or ACE-1-GChM-O) if it would not confuse the user. This choice would account for flipping the data across the appropriate axes, and shifting the longitudinal values from 0^o – 360^o to 180^o -180^o.
- *The user should be able to select a parameter name from a pull-down menu*. Considering that `ecmwfup95110106-95110418.dat` has 76 components, the *Switch* controlling this task must be massive. This *Switch* would then point to the selected data and header file.
- *The program should identify the parameter type*. Before attempting to render the data, the program must know the type or data (two- or three-dimensional, scalar or vector) in order to provide the correct options for the user. This information should be extracted from the header file.
- *The program should establish an interactive control panel*. The parameter type should point to a particular control panel. This control panel, which would include but not be limited to the options seen here, would automatically be displayed on the screen for the user.

Expand analysis capability.

The current scope of the program is in the realm of exploratory visual analysis. However, improvements in analysis capability could make the program more effective. Two ideas are discussed here.

The first idea is to construct an interactive *Probe* with options. This tool would allow the user could click on any point in the image to receive information. This information could be as simple as data value at that point. More complicated tasks would be to make each mouse click show a history of the data value through time or a histogram of all the data, with the particular point highlighted.

Another idea is to animate the development of the data. Using a module such as *Streamlines*, the program would trace the path of a particle instead of merely placing a line on the screen. This method would be extremely useful in analyzing back trajectories. The user could enhance his or her knowledge of the data by establishing an order and flow.

Add a legend.

The programs discussed here may need up to three legends to explain their images to a user: a colored legend, an axis legend, and a vector legend. All of the programs lack at least one of these.

The colored legend appears for every image except those rendered as three-dimensional vector data arrows. For this program, a legend must be constructed to signify that the sign of the w component determines the color of the arrow, either red or blue. This legend would be relatively simple to construct and would take the form of the colored legends seen in other programs.

The axis legend is also necessary to orient the user, particularly when exploring a three-dimensional image. The legend would consist of three unit normal vectors in the u, v, and w directions of the data, and this axis would rotate with the rotating image. As a graphic representation, this idea would strongly reinforce the axis labels, a textual representation that is impossible to read at certain angles. The groundwork for this feature is located in the main directory under `meteo/axes`.

All programs that have arrow representation as a rendering option should also have a vector legend to correlate vector length and magnitude. The legend would have to change when the user zooms in on the image or changes the sampled field. This feature was attempted by using the programs found in the main directory under `meteo/axes`. However, the isolated, single vector in the x-direction was never scaled properly.

Combine parameters.

The current program only permits the user to view one specific parameter set of data at a time. In the future, however, the program should not only accept any parameter, but also let the user render more than one parameter. With this capability, the scientist would not be limited to exploring deposition velocity of a chemical or relative humidity, for example, but could investigate the relationships between the two. This option could cause the image to become overloaded if too many parameters are viewed at once, but the expert scientist should recognize the limits.[6]

Increase rendering speed.

This is a common cry heard among programmers and data analysts alike. Quicker rendering capability, whether through more concise programming or faster visual processing, is necessary for this program to support all of its future tasks. In order for the

[6] McGuinness, 1994.

user to regard the program as an essential exploratory analysis tool, he or she must be able to rotate the image, zoom through it, and make a movie much more quickly than now. The current rendering speed is unacceptable: a sequence of fifteen temperature frames rendered in volume took over one and one half hours to execute. Additional tasks will only decrease the rendering speed. The comprehensive program of the future should not rely on external tools (i.e. QuickTime) to make the images accessible.

CONCLUSIONS

Environmental datasets are too large and complex to study without an analysis tool. A preferred tool is visualization, which can help a user easily detect geographical or temporal patterns unrecognizable in the raw data. The three programs established accomplish this with temperature data, surface wind data, and wind data. Each program also introduces at least one new concept: the temperature program adds a `gif` base map to the scene, the surface wind program presents a method of making a stationary set of axes, and the wind program introduces *Switch* and *Slab* sampling. However, a program that visualizes one set of parameters is impractical: the ultimate goal is to create a program that can handle any parameter from any dataset. For this reason, ideas on future directions of the project are just as crucial as the initial programs themselves. Some suggestions mentioned here are to generalize the program, to expand the analysis capability, to add a legend, to combine parameters, and to increase rendering speed. These ideas will link the three useful yet limited programs to the overall goal of the future: an all-inclusive analytical atmospheric data visualization tool.

REFERENCES

European Center for Medium-Range Weather Forecasts. “Data Coverage Charts.” *European Center for Medium-Range Weather Forecasts (ECMWF)*. 1999. <http://www.ecmwf.int> (27 Jul. 1999).

European Center for Medium-Range Weather Forecasts. “Technical Attachment.” *Description of the ECMWF/WCRP Level III-A Global Atmospheric Data Archive*. 1994. 72 pages.

Fielding, K. Personal Communication to Dr. Carmen Benkovitz. Brookhaven National Laboratory. Upton, NY. 1998.

Gahegan, Mark. Lecture. Penn State University. University Park, PA. 20 Jan. 1999.

IBM Corporation. *IBM Visualization Data Explorer User's Guide*. 6th ed. Yorktown Heights, NY: IBM, 1995.

IBM Corporation. *IBM Visualization Data Explorer User's Reference*. 3rd ed. Yorktown Heights, NY: IBM, 1995.

IBM Visualization Data Explorer Tutorial Tape Set (NTSC). Version 2. Tape 1 of 2, *Creating Visual Programs*. IBM, no year.

McGuinness, Carol. "Expert/Novice Use of Visualization Tools." In MacEachren, A. M. and D. R. F. Taylor, *Visualization in Modern Cartography*. Vol.2. Tarrytown, NY: Elsevier, 1994.

ACKNOWLEDGEMENTS

The author would like to extend deepest thanks to Dr. Carmen Benkovitz of the Environmental Chemistry Division (ECD) and Dr. Mike McGuigan of the Information Technology Division (ITD) of the Brookhaven National Laboratory (BNL). Their expertise and guidance has made this part of the project possible. Others who contributed include Mr. Altaf Mubaraki, Mr. Ernie Lewis and Dr. Fred Brechtel of ECD, and Dr. Gordon Smith and Mr. Ron Oglesby from ITD. A special thanks goes out to the coordinators of the Summer Undergraduate Research Experience (SURE), Dr. Jeffrey Gaffney of Argonne National Laboratory and Dr. Milton Constantine of Oak Ridge National Laboratory, and all of the student colleagues there, especially Anthony Nguyen of the University of Oklahoma.

APPENDICES

- Appendix A: Meteorological Data Parameters
- Appendix B: File Structure
- Appendix C: Program Pages
- Appendix D: Making a Movie

APPENDIX A: Meteorological Data Parameters

This table gives the 76 parameters that can be found in the `ecmwfup95110106-95110418.dat` file. Each entry shows the parameter number, name, and units. See the *Technical Attachment* reference for the source of this table.

129 geopotential (m^2/s^2)
130 temperature (K)
131 u-component of wind (m/s)
132 v-component of wind (m/s)
133 specific humidity (kg/kg)
134 surface pressure (Pa)
135 vertical velocity (Pa/s)
138 vorticity (1/s)
139 surface temperature (K)
140 soil moisture (m)
141 snow depth (m)
142 large scale precipitation (m)
143 convective precipitation (m)
146 surface sensible heat flux (W/m^2)
147 surface latent heat flux (W/m^2)
151 mean sea level pressure (Pa)
152 ln surface pressure (Pa)
155 divergence (1/s)
164 total cloud cover (0-1)
165 u-wind at 10 m (m/s)
166 v-wind at 10 m (m/s)
167 temperature at 2 m (K)
168 dew point at 2 m (K)
173 surface roughness (m)
176 surface net shortwave radiation (W/m^2)
177 surface net longwave radiation (W/m^2)
178 TOA net shortwave radiation (W/m^2)
179 TOA net longwave radiation (W/m^2)
182 evaporation (m)
185 convective cloud cover (0-1)
186 low cloud cover (0-1)
187 medium cloud cover (0-1)
188 high cloud cover (0-1)
203 condensed moisture (mol/mol)
246 convective cloud water content (kg/kg)
248 cloud fraction at vertical levels (0-1)
500 vertical velocity (1/s)
501 roughness length from Wesely (m)
503 condensed moisture (mol/mol)
505 vertical diffusivity coeff for heat (m^2/s)
506 vertical diffusivity coefficient for momentum (m^2/s)
507 index to 1st ECMWF level above PBL
508 relative humidity (%)
510 friction velocity (m/s)
511 deposition velocity, SO_2 (m/s)
512 deposition velocity, H_2O_2 (m/s)
513 deposition velocity, SO_4 (m/s)
514 deposition velocity, O_3 (m/s)
515 deposition velocity, NO_2 (m/s)
516 deposition velocity, NO (m/s)
517 deposition velocity, HNO_3 (m/s)
518 deposition velocity, ALD (m/s)
519 deposition velocity, HCHO (m/s)
520 deposition velocity, OP (m/s)
521 deposition velocity, HNO_2 (m/s)
522 deposition velocity, PAA (m/s)
523 deposition velocity, ORA (m/s)
524 deposition velocity, NH_3 (m/s)
525 deposition velocity, PAN (m/s)
530 mixing ratio, OH
531 mixing ratio, HO_2
532 mixing ratio, H_2O_2
533 mixing ratio, O_3
534 calculated PBL height (m)
600 potential temperature (K)
601 virtual potential temperature (K)
602 equivalent potential temperature (K)
603 saturation potential temperature (K)
610 convective cloud base index (1-#lvls)
611 convective cloud top index (1-#lvls)
612 convective mixing parameter (0-1)
613 cloud temperature (K)
614 DMS emissions flux ($mol/cm^2/s$)
IVCLD convective cloud fraction at vertical levels (0-1)
1133 upper air wind speed & dir (m/s, ^{o})
1167 wind speed & dir at 10 m (m/s, ^{o})

APPENDIX B: File Structure

Here are some pointers to remember before attempting to locate the files in this file list:

- They can be found under `/usr2/people/aliszka` on **`mary`**.
- Bolded words signify directories, while indentations may signify subdirectories or actual files.
- Numbers in parentheses mean that more than one file of that name exists. The other files can be found by adding or changing the listed file by those numbers.
- Where a `net` file is listed the associated `cfg` is also present.

```
examples
    images
        scalar3D
            I12_diag_z1.tiff                V_offbot_z3.tiff
            I12_offbot_z3.tiff              V_offdiag_z5.tiff
            I4_offdiag_z5.tiff              V_offfront_z3.tiff
            V_diag_z1.tiff
        vector2D
            A00403.tiff                     S00606.tiff
            A00606.tiff                     S01209.tiff
            C.tiff                          S03009.tiff
            N.tiff                          T12081.tiff
            R03618_2.tiff
        vector3D
            Adiag03618-9.tiff               Aofffront00606-9.tiff
            Aoffdiag00909-3.tiff            Vdiag-9.tiff
    movies
        1
        a.out
        mkmv.c
        mkmv_15
        scalar3D
            I10_z3.mov                      V.mov
            I12.mov                         V_offbot.mov
            I6.mov
        vector2D
            A00403.mov                      S01209.mov
            A00606.mov                      S03009.mov
            C.mov                           T12081.mov
            R03618_2.mov
        vector3D
            A00909-3.mov                    Vdiag-9.mov
            Aofffront00606-9.mov
headers
    newtemp
        timetemp.dx                         vnitemp.dx
    newwind
        timewind.dx                         vniVwind.dx
        vniUwind.dx                         vniWwind.dx
    old
        hadron_vni_list.dx                  time.dx
        header2.general
```

- **maps**
  - N180.gif
  - N360.gif
  - S180.gif
  - S360.gif
  - **refmaps**
    - bettermap.gif
    - hein_world.gif
    - **withgrid**
      - N180grid.gif
      - N360grid.gif
      - S180grid.gif
      - S360grid.gif
- **meteo**
  - **axes**
    - Legend.net
    - axes2.dx
    - axes3.cfg
  - insert_map.net
  - **scalar3D**
    - geo_temp.net (312,32)
  - **scalar3Danim**
    - tempanim.net
  - **scalar3Ddata**
    - geo.dx
    - geo.filter
    - geo.general
    - temp95110106.dat (112,118,200,212,218,300,306,312,318,400,406, 412,418)
  - **vector2D**
    - surface.net (3)
  - **vector2Danim**
    - preSWA.net
    - surfacewindanim.net
  - **vector3Danim**
    - windanim.net (2)
  - **vectordata**
    - uwind106.general
    - uwind95110106.dat (112,118,200,212,218,300,306,312,318,400,406, 412,418)
    - vwind106.general
    - vwind95110106.dat (112,118,200,212,218,300,306,312,318,400,406, 412,418)
    - wwind106.general
    - wwind95110106.dat (112,118,200,212,218,300,306,312,318,400,406, 412,418)
- **samples**
  - elevation.general
  - quark_aps6.net
  - sample2.net (3)
- **textfiles**
  - carmeninfo.txt
  - filelist.txt
  - makemovie.pro
  - pvwave.pro

APPENDIX C.1: Program Pages for Three-Dimensional Scalar Data

APPENDIX C.2: Program Pages for Two-Dimensional Vector Data

APPENDIX C.3: Program Pages for Three-Dimensional Vector Data

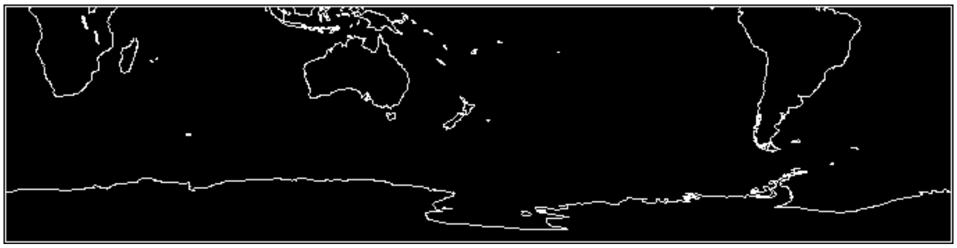
APPENDIX D: Making a Movie

In order to construct a QuickTime movie, you must have the player installed on your workstation. Once you have the player, follow these steps:

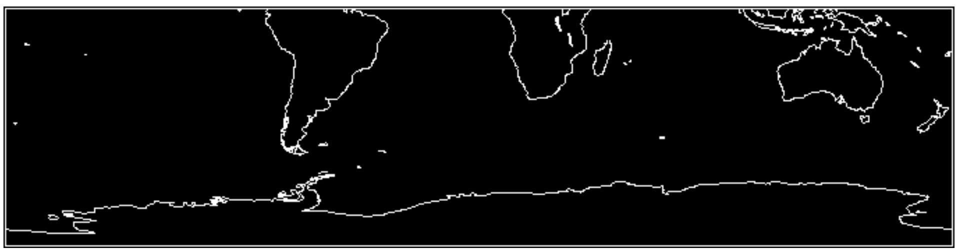
- Acquire the files `mkmv.c` and `mkmv_15`. In the main directory, they are found under the `examples/movies`.
- Check the image path in `mkmv.c` and the parameters in `mkmv_15`. If they are not correct, change them. Remember that the length and width of the image window must both be divisible by four in order to use QuickTime
- Open a UNIX shell.
- If you changed anything in `mkmv_15`, type the command `chmod 750 mkmv_15`.
- Type `cc mkmv.c`.
- Type `a.out>1`.
- Type `mkmv_15 1`. This step should take awhile.
- Type `movieplayer movie.mov` and watch the movie.
- If you wish to keep the movie, rename it so it doesn't get written over.

APPENDIX E: Available Maps

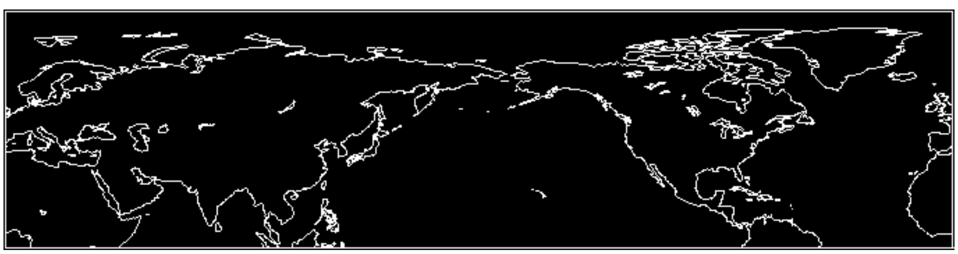
The project only used data from the Southern Hemisphere that was taken on the 0-360^o longitudinal scale. In the future, however, the user should have the choice of viewing Northern or Southern Hemisphere data from 0^o - 360^o or from 180^o - 180^o. For this reason, several base maps were made from the data cover map given on ECMWF web page. Each map below is available under `maps` in the main directory.



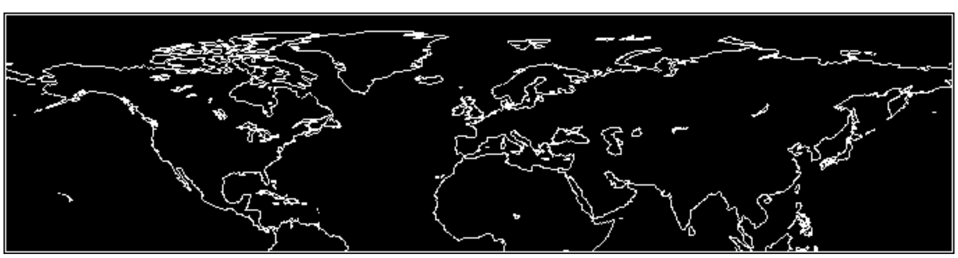
The map currently used. Southern Hemisphere, 0^o - 360^o. (`S360.gif`)



Southern Hemisphere, 180^o - 180^o. (`S180.gif`)



Northern Hemisphere, 0^o - 360^o. (`N360.gif`)



Northern Hemisphere, 180^o - 180^o. (`N180.gif`)